US009485291B2

(12) **United States Patent**
Shetty et al.

(10) **Patent No.: US 9,485,291 B2**
(45) **Date of Patent: Nov. 1, 2016**

(54) IMPARTING REAL-TIME PRIORITY-BASED NETWORK COMMUNICATIONS IN AN ENCRYPTED COMMUNICATION SESSION

(71) Applicant: **ADOBE SYSTEMS INCORPORATED**, San Jose, CA (US)

(72) Inventors: **Pritham Shetty**, Los Altos, CA (US); **Asa Whillock**, San Francisco, CA (US); **Edward Chan**, Fremont, CA (US); **Srinivas Manapragada**, Fremont, CA (US); **Matthew Kaufman**, Bonny Doon, CA (US); **Michael Thornburgh**, San Jose, CA (US)

(73) Assignee: **ADBOE SYSTEMS INCORPORATED**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **14/579,801**

(22) Filed: **Dec. 23, 2014**

(65) **Prior Publication Data**

US 2015/0142927 A1 May 21, 2015

Related U.S. Application Data

(63) Continuation of application No. 13/584,484, filed on Aug. 13, 2012, now Pat. No. 8,918,644, which is a continuation of application No. 13/281,317, filed on Oct. 25, 2011, now Pat. No. 8,245,033, which is a continuation of application No. 12/252,357, filed on Oct. 15, 2008, now Pat. No. 8,051,287.

(51) **Int. Cl.**

H04L 29/06	(2006.01)
H04L 9/32	(2006.01)
H04L 29/08	(2006.01)

(52) **U.S. Cl.**

CPC ***H04L 65/60*** (2013.01); ***H04L 9/3242*** (2013.01); ***H04L 9/3271*** (2013.01); ***H04L 63/0869*** (2013.01); ***H04L 67/32*** (2013.01); *H04L 63/06* (2013.01)

(58) **Field of Classification Search**

CPC H04L 9/002; H04L 9/08; H04L 9/0819
USPC 713/168
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

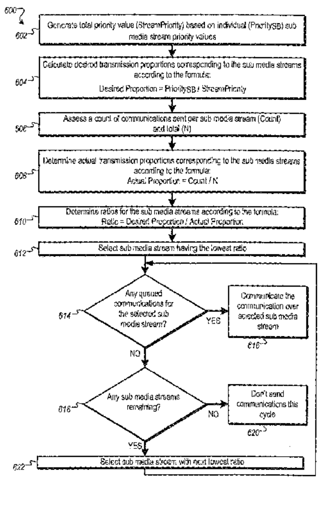
2002/0156913 A1*	10/2002	Tsang	H04J 3/1617 709/236
2008/0253466 A1*	10/2008	Fu	H04N 21/2389 375/240.26
2009/0232084 A1*	9/2009	Li	H04L 5/0007 370/330

Primary Examiner — Mohammad W Reza
(74) *Attorney, Agent, or Firm* — Shook Hardy & Bacon L.L.P.

(57) ABSTRACT

This specification describes technologies relating to imparting real-time priority-based network communications in an encrypted session. In general, aspects of the subject matter described can be embodied in methods that include establishing, based on cryptographic information in a reserved, random-data portion of a handshake communication, a session, receiving parameter values relating to a sub media stream, included in a header of a network communication, storing the parameter values, obtaining state information and a data payload included in a second network communication, identifying, from the state information, a purpose of the second network communication, and whether a header of the second network communication includes one or more new values corresponding to one or more of the parameters, updating one or more of the stored values based on the one or more new values, and processing the data payload based on the identified purpose and the stored parameter values.

20 Claims, 10 Drawing Sheets


600
602
Generate total priority value (StreamPriority) based on individual (PrioritySB) sub media stream priority values
604
Calculate desired transmission proportions corresponding to the sub media streams according to the formula:
Desired Proportion = PrioritySB / StreamPriority
506
Assess a count of communications sent per sub media stream (Count) and total (N)
608
Determine actual transmission proportions corresponding to the sub media streams according to the formula:
Actual Proportion = Count / N
610
Determine ratios for the sub media streams according to the formula:
Ratio = Desired Proportion / Actual Proportion
612
Select sub media stream having the lowest ratio
614
Any queued communications for the selected sub media stream?
YES
Communicate the communication over selected sub media stream
616
NO
618
Any sub media streams remaining?
NO
Don't send communications this cycle
620
YES
622
Select sub media stream with next lowest ratio

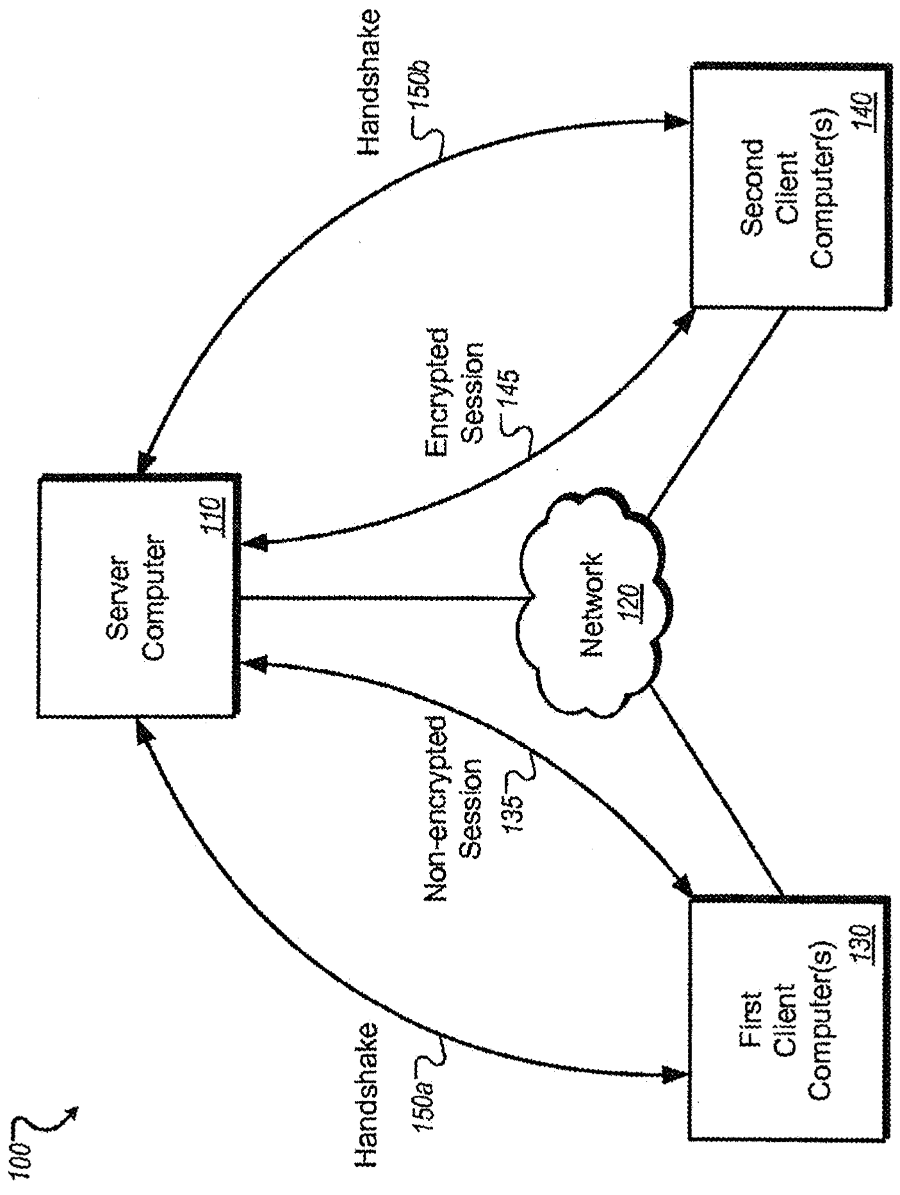

100
Handshake
150a
Non-encrypted Session
135
Server Computer
110
Encrypted Session
145
Handshake
150b
Network
120
First Client Computer(s)
130
Second Client Computer(s)
140

FIG. 1

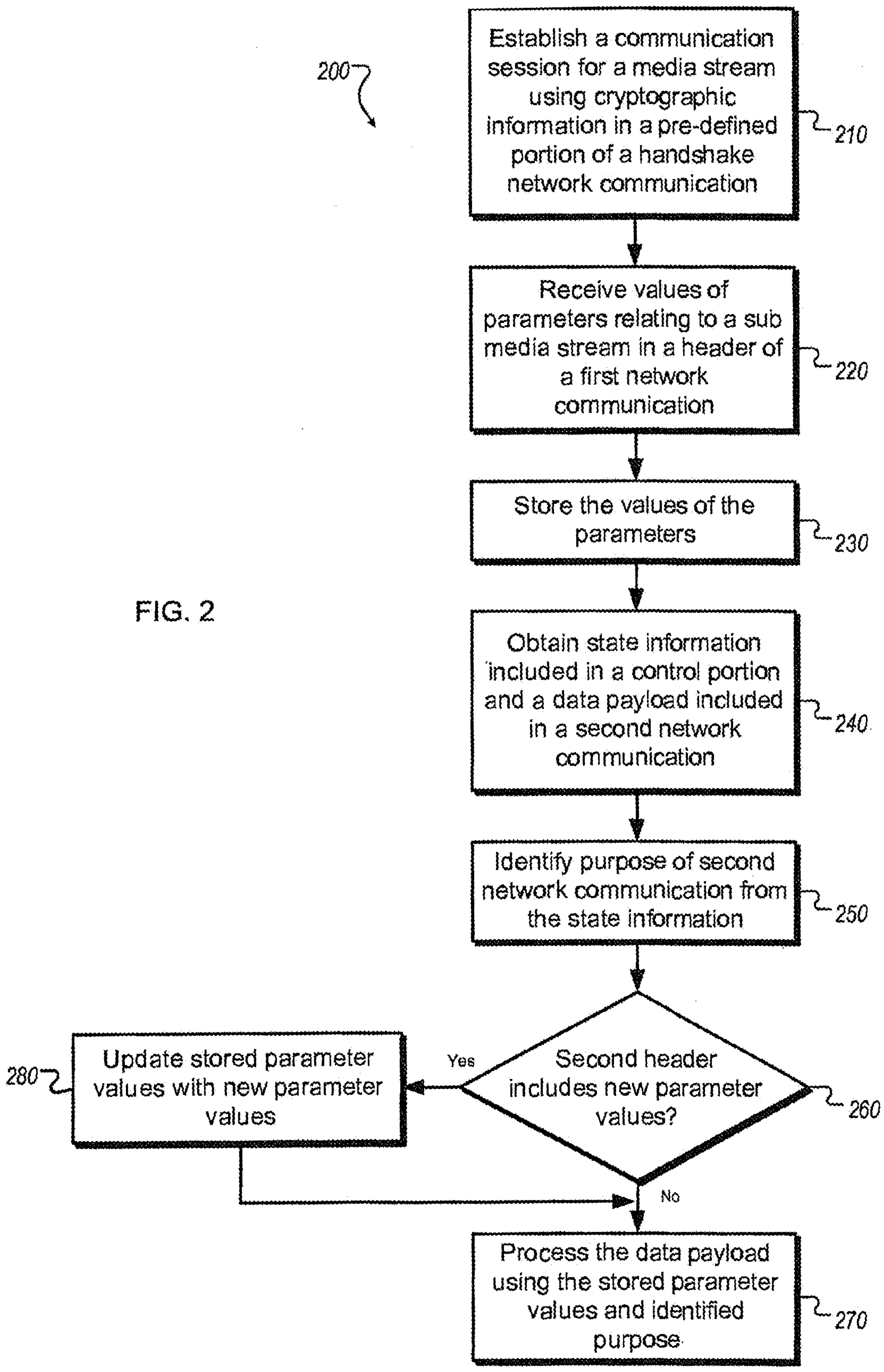

200
Establish a communication session for a media stream using cryptographic information in a pre-defined portion of a handshake network communication
210
Receive values of parameters relating to a sub media stream in a header of a first network communication
220
Store the values of the parameters
230
Obtain state information included in a control portion and a data payload included in a second network communication
240
Identify purpose of second network communication from the state information
250
Second header includes new parameter values?
260
Yes
No
Update stored parameter values with new parameter values
280
Process the data payload using the stored parameter values and identified purpose
270

FIG. 2

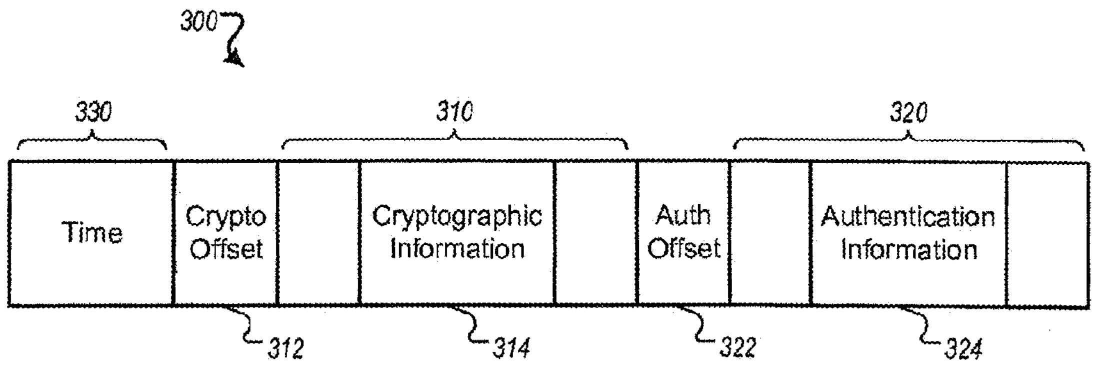


FIG. 3A

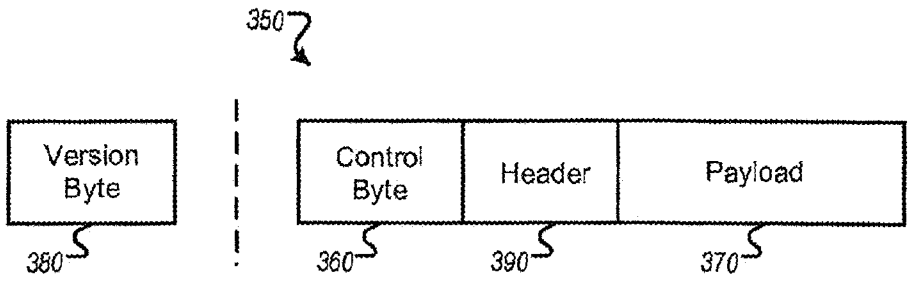


FIG. 3B

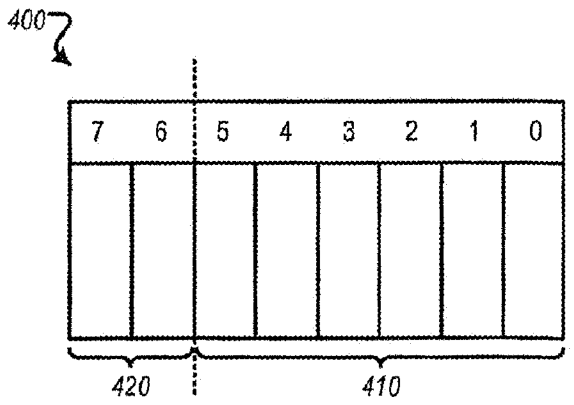
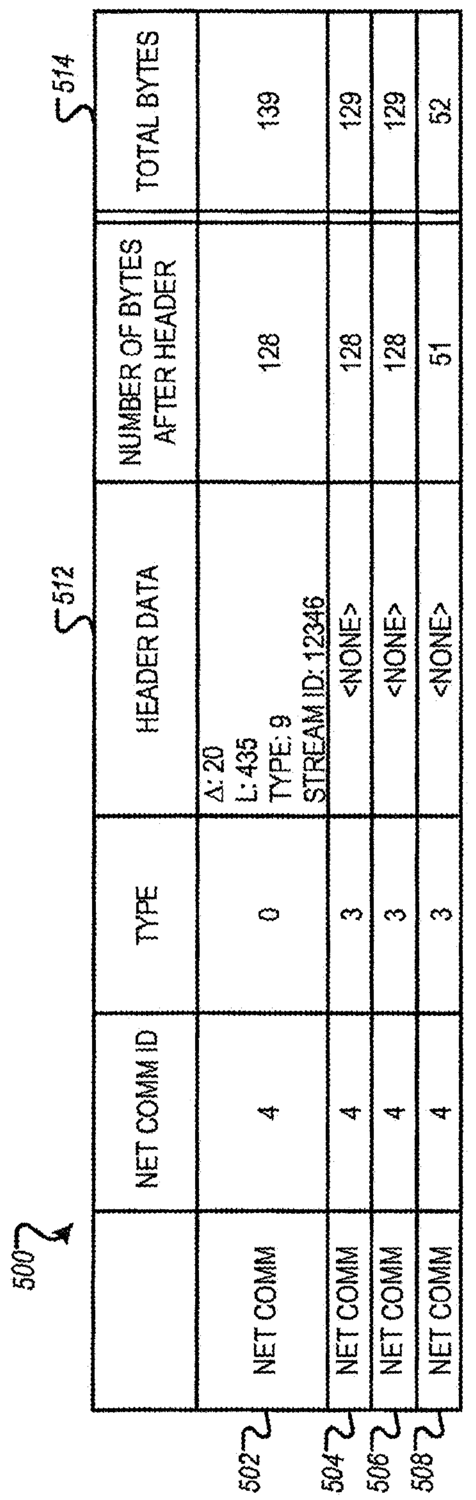


FIG. 4


500
502
504
506
508
512
514

	NET COMM ID	TYPE	HEADER DATA	NUMBER OF BYTES AFTER HEADER	TOTAL BYTES
NET COMM	4	0	Δ: 20 L: 435 TYPE: 9 STREAM ID: 12346	128	139
NET COMM	4	3	<NONE>	128	129
NET COMM	4	3	<NONE>	128	129
NET COMM	4	3	<NONE>	51	52

FIG. 5A

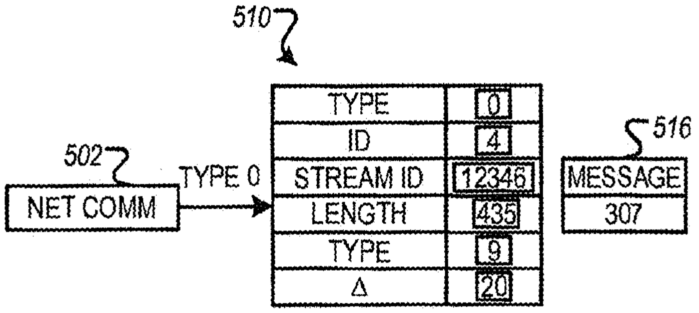


FIG. 5B

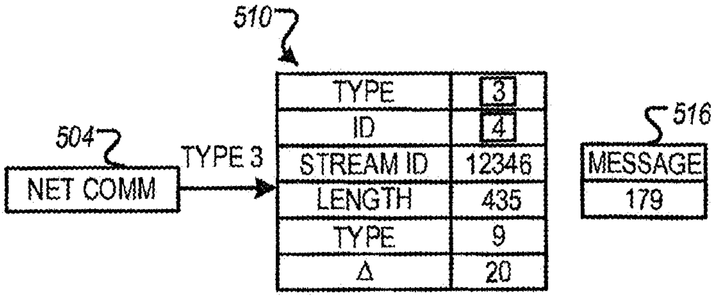


FIG. 5C

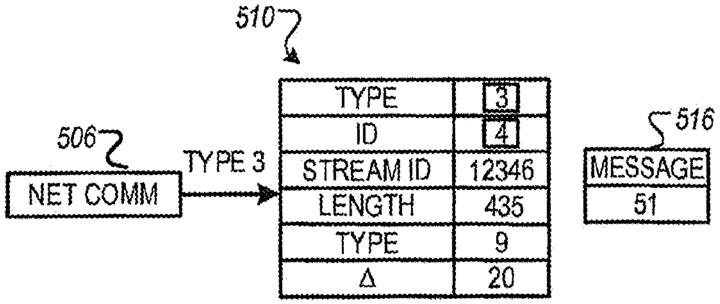


FIG. 5D

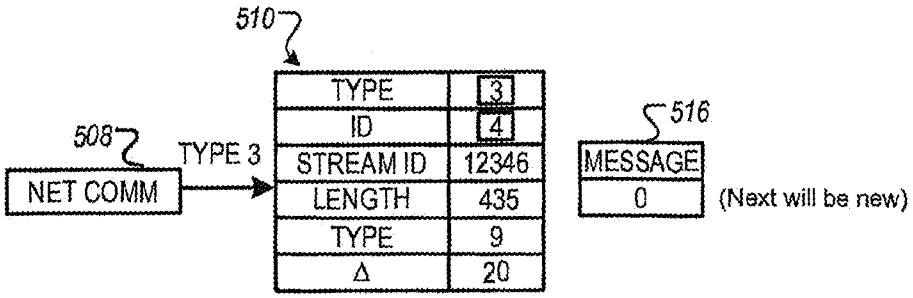


FIG. 5E

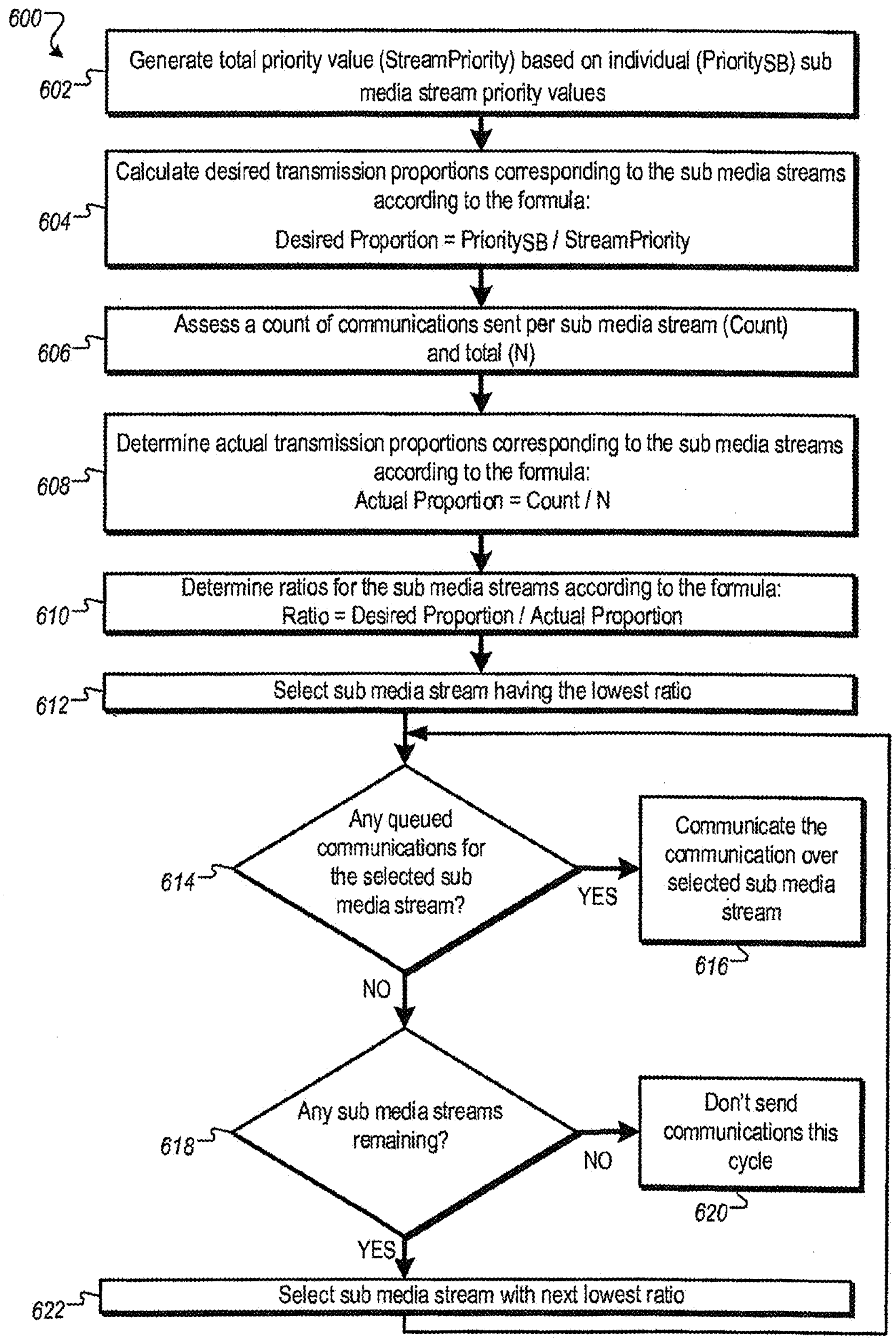

600
602
Generate total priority value (StreamPriority) based on individual ($Priority_{SB}$) sub media stream priority values
604
Calculate desired transmission proportions corresponding to the sub media streams according to the formula:
Desired Proportion = $Priority_{SB}$ / StreamPriority
606
Assess a count of communications sent per sub media stream (Count) and total (N)
608
Determine actual transmission proportions corresponding to the sub media streams according to the formula:
Actual Proportion = Count / N
610
Determine ratios for the sub media streams according to the formula:
Ratio = Desired Proportion / Actual Proportion
612
Select sub media stream having the lowest ratio
614
Any queued communications for the selected sub media stream?
YES
Communicate the communication over selected sub media stream
616
NO
618
Any sub media streams remaining?
NO
Don't send communications this cycle
620
YES
622
Select sub media stream with next lowest ratio

FIG. 6

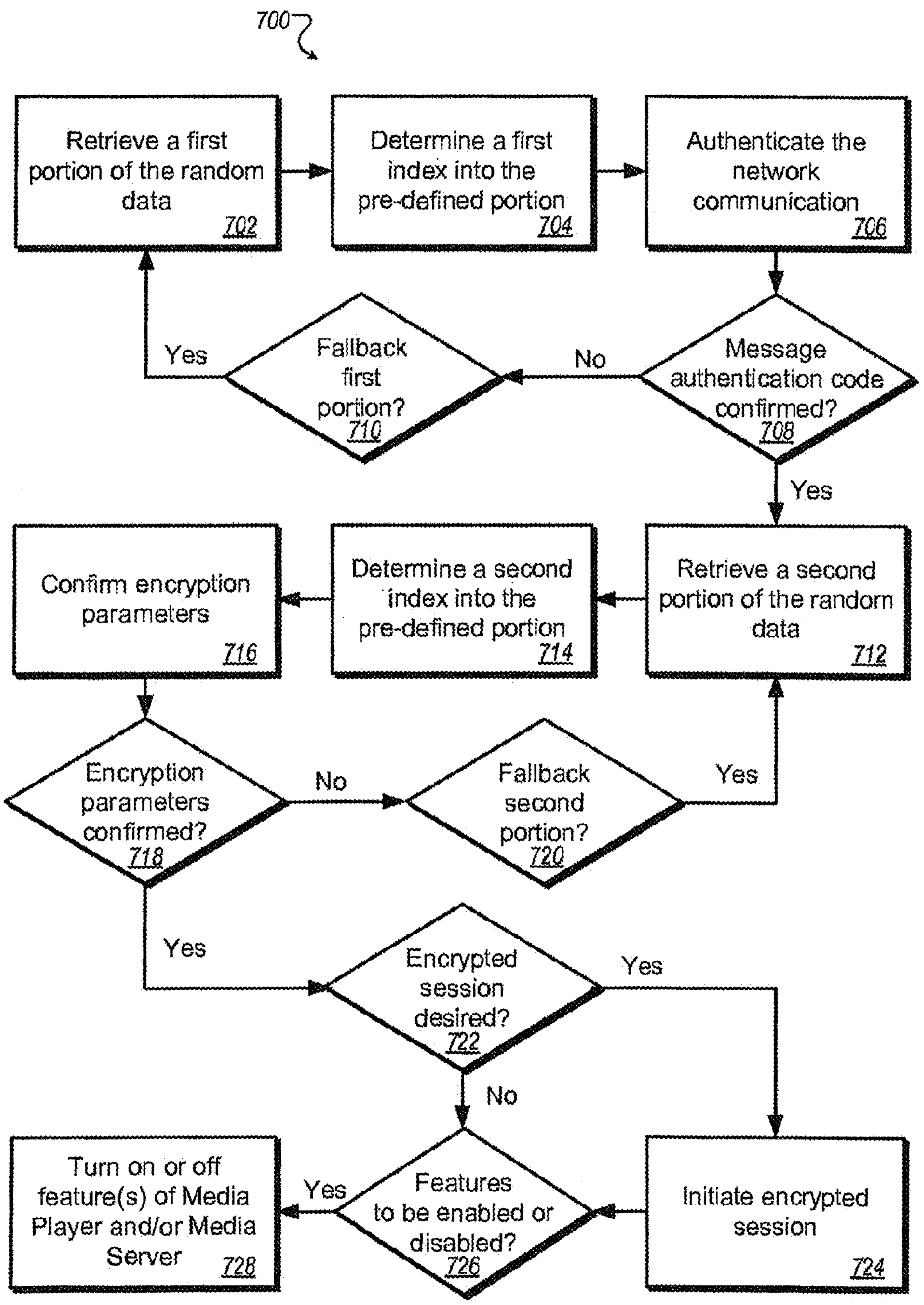

700
Retrieve a first portion of the random data 702
Determine a first index into the pre-defined portion 704
Authenticate the network communication 706
Message authentication code confirmed? 708
No
Fallback first portion? 710
Yes
Yes
Retrieve a second portion of the random data 712
Determine a second index into the pre-defined portion 714
Confirm encryption parameters 716
Encryption parameters confirmed? 718
No
Fallback second portion? 720
Yes
Yes
Encrypted session desired? 722
Yes
No
Initiate encrypted session 724
Features to be enabled or disabled? 726
Yes
Turn on or off feature(s) of Media Player and/or Media Server 728

FIG. 7

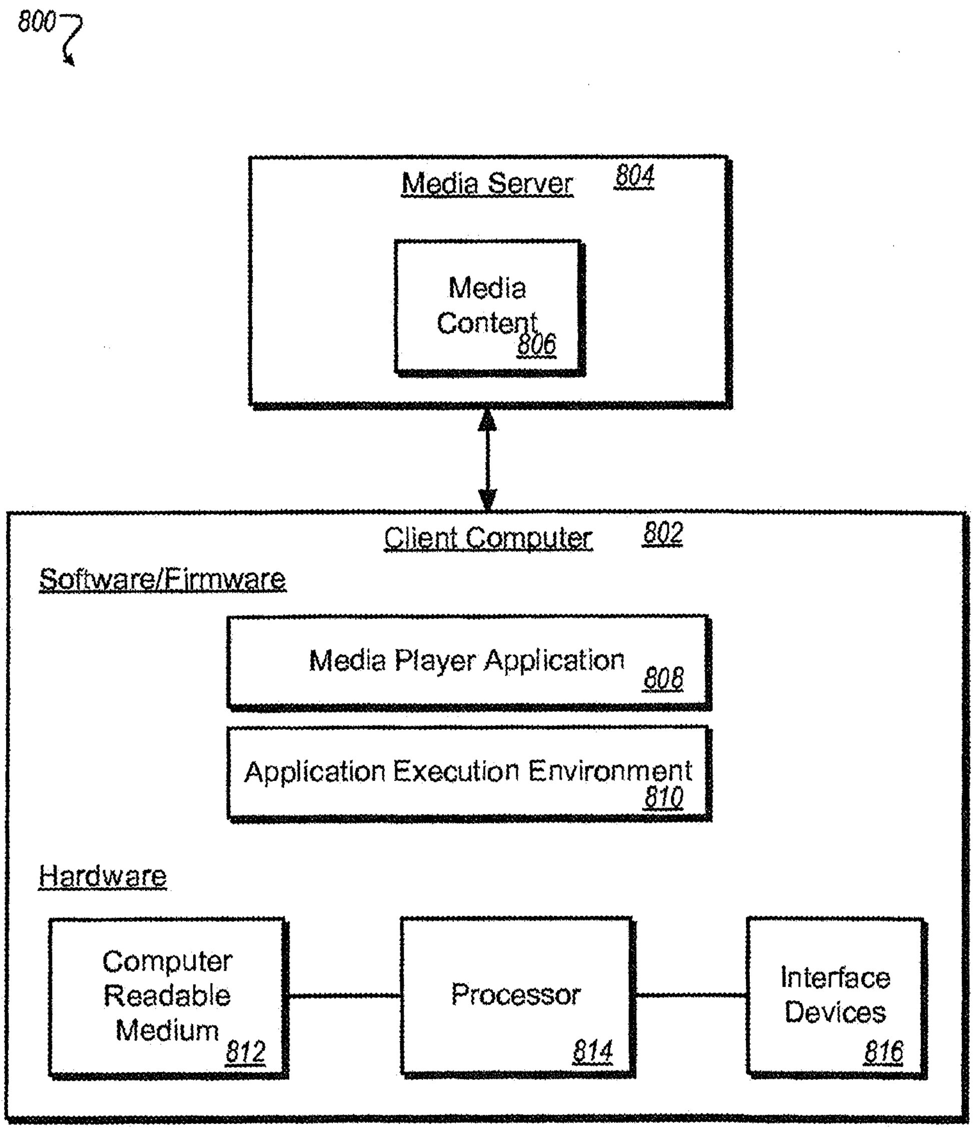

800
Media Server 804
Media Content 806
Client Computer 802
Software/Firmware
Media Player Application 808
Application Execution Environment 810
Hardware
Computer Readable Medium 812
Processor 814
Interface Devices 816

FIG. 8

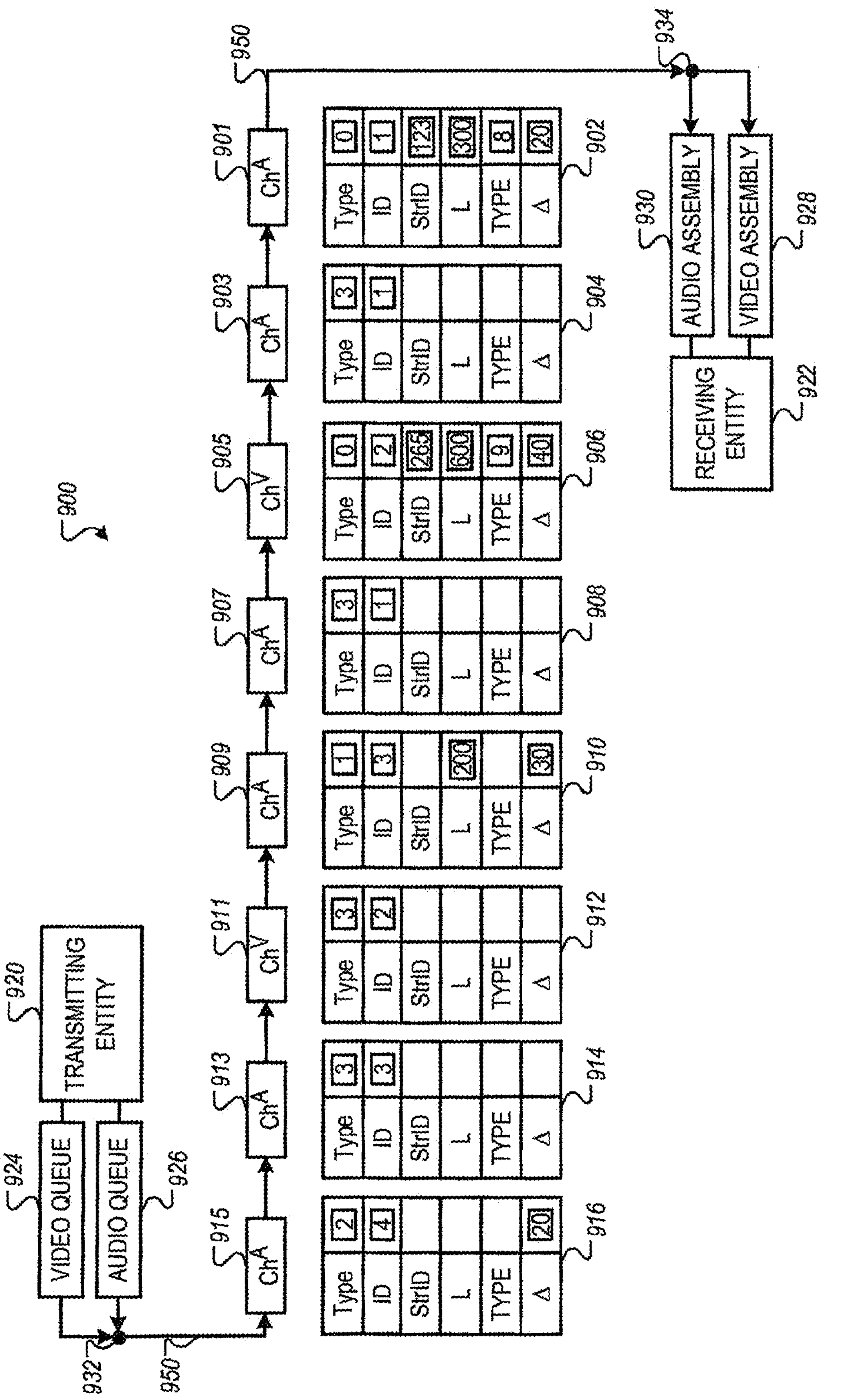

900
TRANSMITTING ENTITY
920
VIDEO QUEUE
924
AUDIO QUEUE
926
932
950
Ch^A
915
Ch^A
913
Ch^V
911
Ch^A
909
Ch^A
907
Ch^V
905
Ch^A
903
Ch^A
901
950
934
AUDIO ASSEMBLY
930
VIDEO ASSEMBLY
928
RECEIVING ENTITY
922
Type 2, ID 4, StrID, L, TYPE, Δ 20
916
Type 3, ID 3, StrID, L, TYPE, Δ
914
Type 3, ID 2, StrID, L, TYPE, Δ
912
Type 1, ID 3, StrID, L 200, TYPE, Δ 30
910
Type 3, ID 1, StrID, L, TYPE, Δ
908
Type 0, ID 2, StrID 265, L 600, TYPE 9, Δ 40
906
Type 3, ID 1, StrID, L, TYPE, Δ
904
Type 0, ID 1, StrID 123, L 300, TYPE 8, Δ 20
902

FIG. 9

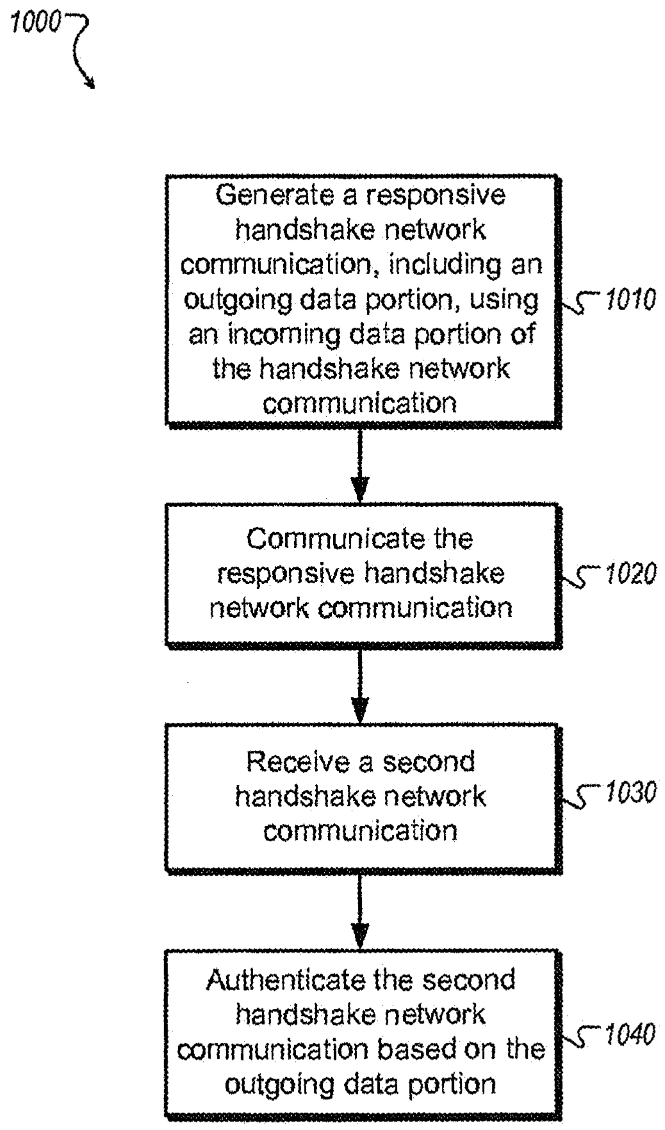

1000
Generate a responsive handshake network communication, including an outgoing data portion, using an incoming data portion of the handshake network communication
1010
Communicate the responsive handshake network communication
1020
Receive a second handshake network communication
1030
Authenticate the second handshake network communication based on the outgoing data portion
1040

FIG. 10

IMPARTING REAL-TIME PRIORITY-BASED NETWORK COMMUNICATIONS IN AN ENCRYPTED COMMUNICATION SESSION

PRIORITY CLAIM

This patent document is a continuation, and claims benefits of priority under 35 USC 120, of U.S. application Ser. No. 13/584,484, filed Aug. 13, 2012, which is a continuation of U.S. application Ser. No. 13/281,317, filed Oct. 25, 2011, which is a continuation of U.S. application Ser. No. 12/252,357, filed Oct. 15, 2008, all of which are incorporated herein by reference in their entirety.

BACKGROUND

The present disclosure relates to imparting real-time priority-based network communications in an encrypted communication session.

A computer network is a collection of processing nodes coupled together with wired and/or wireless communications links. A computer network can be a single network or a collection of networks (e.g., an internetwork), and can use multiple networking protocols, including internetworking protocols (e.g., Internet Protocol (IP)). These protocols define the manner in which information is prepared for transmission through the network, and typically involve breaking data into segments genetically known as packets (e.g., IP packets, ATM (Asynchronous Transfer Mode) cells) for transmission. These networking protocols are typically organized by a network architecture having multiple layers, where a layer provides communication services to the layer above it. The protocols can include shared-line protocols such as in Ethernet networks, connection-oriented switching protocols such as in AIM networks, and/or connectionless packet-switched protocols such as in IP.

Many computer networks use connectionless packet-switched protocols (e.g., IP). Packets are routed separately and can thus take different paths through the network. Various protocols have been built on top of connectionless packets witched protocols, such as IP, to provide connection based communications over the underlying connectionless protocol. For example, Transmission Control Protocol (TCP), of the TCP/Internet Protocol (IP) protocol stack, sets up a connection at both ends of a communications and attaches a header to the packets that contains the source and destination ports as well as the sequence number of the packet and other administrative information. The destination typically receives a number of TCP packets before sending an acknowledgement to the source. If the acknowledgment fails, the source will generally presume the packets were lost and retransmit those packets. This process ensures reliable delivery.

Another protocol, UDP is connectionless and unreliable, meaning that the protocol does not establish a connection at both ends and does not include a resource for resending lost packets. Real-time Transport Protocol (RTP) operates as a thin layer on top of UDP to create a generalized multipurpose real-time transport protocol. RTP provides support for applications with real-time properties including reconstruction, loss detection, security, and content identification without reliability-induced delays associated with TCP. Real-Time Control Protocol (RTCP) works in conjunction with RTF to provide control support to an application for maintaining an RTP session.

The interactions of various aspects of networks can be complex. Many applications use RTP for transporting media streams. RTP is built over UDP, which generally has no fixed ports associated with it. Firewalls can prevent unauthorized and/or hostile access to a computer system or network. As such, firewalls are generally configured with strict rules specifying static ports through which desired and/or authorized data traffic can pass, while blocking undesirable data. There is no guarantee that a port associated with an incoming RTP/UDP stream will be allowed through the firewall. In addition, NAT devices, which can allow a network to support many more clients than it has fixed IP addresses, are used to translate an IP address used within one network to a different IP address known within another network.

SUMMARY

This specification describes technologies relating to imparting real-time priority-based network communications in an encrypted communication session. In general, one aspect of the subject matter described in this specification can be embodied in a method that includes establishing, based at least in part on cryptographic information in a pre-defined portion of a handshake network communication, a communication session to communicate a media stream, where the pre-defined portion of the handshake network communication is reserved for random data, receiving through the communication session, as part of the media stream, values of parameters relating to a sub media stream, included in a first header portion of a first real-time, priority-based network communication, storing the values of the parameters, obtaining through the communication session, as part of the media stream, state information included in a control portion of a second real-time, priority-based network communication and a data payload included in the second network communication, identifying, from the state information, a purpose of the second network communication in relation to the media stream, and whether a second header portion of the second network communication includes one or more new values corresponding to one or more of the parameters, updating, when the second header portion includes the one or more new values, one or more of the stored values based at least in part on the one or more new values, and processing the data payload based at least in part on the identified purpose and the stored values of the parameters.

In addition, another aspect of the described subject matter can be embodied in a method, where the sub media stream is a first sub media stream, that includes receiving through the encrypted communication session, as part of the media stream, real-time, priority-based network communications associated with a second sub media stream, where the network communications include state information and data payloads, decrypting the data payloads based at least in part on the cryptographic information, assembling a playable portion of the second sub media stream based at least in part on state information and the decrypted data payloads, and playing the assembled portion of the second sub media stream relative to a media type indicated by at least one of the stored values of the parameters. The first and second sub media streams can be the same sub media stream in some embodiments (e.g., both the video portion of a media stream) while in others the sub media streams are distinct (e.g., video and audio portions, respectively, of a media stream). Another aspect of the described subject matter can be embodied in a method that includes turning on or off a feature of a program operating in relation to the communication session based at least in part on the establishing of the

communication session. Other embodiments of these aspects include corresponding systems, apparatuses, and computer program products.

These and other embodiments can optionally include one or more of the following features. Identifying the purpose can include identifying the purpose from a set of possible Purposes including beginning a new sub media stream, the second header portion including sub media stream values corresponding to the parameters, indicating a new media type, the second header portion including the one or more values corresponding to fewer than all of the parameters, providing a new time, the second header portion including one new value corresponding to one of the parameters, and continuing the sub media stream, the second header portion not including new parameter values.

Establishing the communication session can include generating, using an incoming data portion from the handshake network communication, a responsive handshake network communication, including an outgoing data portion where the outgoing data portion includes at least one of random data and cryptographic information, and the incoming data portion includes at least one of random data and cryptographic information in the handshake network communication, communicating the generated responsive handshake network communication, receiving a second handshake network communication in response to the communicated responsive handshake network communication, and authenticating the second handshake network communication based on the outgoing data portion in the communicated responsive handshake network communication. In addition, establishing the communication session can include establishing a cryptographic key based at least in part on the cryptographic information, and initiating the communication session as an encrypted communication session based at least in part on the cryptographic key.

The described subject matter can also be embodied in a method that includes establishing, based at least in part on cryptographic information in a pre-defined portion of a handshake network communication, a communication session to communicate a media stream, where the pre-defined portion of the handshake network communication is reserved for random data and the media stream includes sub media streams, associating priority values with the sub media streams, generating a total priority value based on the priority values, calculating desired transmission proportions corresponding to the sub media streams based on the priority values and the total priority value, communicating a first set of real-time priority based network communications over the sub media streams, assessing sub media stream counts corresponding to the sub media streams based on the first set of network communications communicated in the sub media streams, summing a total count based on the sub media stream counts, determining actual transmission proportions corresponding to the sub media streams based on the sub media stream counts and the total count, figuring ratios corresponding to the sub media streams based on the desired transmission proportions and the actual transmission proportions, and storing the figured ratios. In addition, another aspect of the described subject matter can be embodied in a method drat includes selecting a low-ratio sub media stream corresponding to a lowest of the ratios, queuing a second set of network communications to be communicated over the sub media streams, picking a low-ratio sub media stream network communication, associated with the low-ratio sub media stream, from the queued second set of network

communications, and communicating the low-ratio sub media stream network communication over the low-ratio sub media stream.

According to another aspect, the described subject matter can also be embodied in a system including a server, programmed to establish encrypted sessions over a network with a client using a session startup handshake including a handshake network communication including a pre-defined portion reserved for random data, and the client, programmed to establish encrypted sessions with the server using the session startup handshake including cryptographic information in the pre-defined portion of the handshake network communication, where the server and the client are programmed to perform operations including establishing, based at least in part on the cryptographic information, an encrypted communication session to communicate a media stream, receiving through the communication session, as part of the media stream, values of parameters relating to a sub media stream, included in a first header portion of a first real-time, priority-based network communication, storing the values of the parameters, obtaining through the communication session, as part of the media stream, state information included in a control portion of a second real-time, priority-based network communication and a data payload included in the second network communication, identifying, from the state information, a purpose of the second network communication in relation to the media stream, and whether a second header portion of the second network communication includes one or more new values corresponding to one or more of the parameters, updating, when the second header portion includes the one or more new values, one or more of the stored values based at least in part on the one or more new values, and processing the data payload based at least in part on the identified purpose and the stored values of the parameters. The client can be a mobile device (such as a mobile phone, game machine, personal digital assistant, and laptop computer) and a stationary device (such as a workstation, desktop computer, and super computer). The operations can include the various operations of the method(s) described.

In another aspect, the described subject matter can also be embodied in a method including dividing sub media streams, making up media streams, into network communications, assigning states to the network communications representative of information regarding the associated sub media streams and the associated media streams, writing the states into headers of the network communications, queuing the network communications for transmission, and transmitting some of the network communications according to a scheduling process. Another aspect of the described subject matter can be embodied in a method that includes receiving the transmitted network communications, translating the states associated with the network communications into information regarding the associated sub media streams and the associated media streams, and reassembling the network communications into the associated sub media streams for replay of the media streams. In addition, an aspect can be embodied in a method that includes assigning a priority to each sub media stream, totaling a priority for each media stream, calculating a desired transmission proportion for each sub media stream using the assigned priority and the totaled priority, counting a sub-total of network communications transmitted per the sub media stream during a preset period, counting a total number of network communications transmitted during the preset period, calculating an actual transmission proportion for each sub media stream using the sub total and the total, calculating a ratio between the desired transmission proportion and the actual transmission proportion, determining a sub media stream having a lowest ratio, checking a queue of the network communications for a network communication associated with the determined sub media stream, and transmitting the network communication associated with the determined sub media stream when found in the queue.

These and other embodiments can optionally include one or more of the following features. The states can indicate possible purposes of a corresponding network communication, and the possible purposes can include a new stream network communication associated with a new media message sub media stream where the new stream network communication includes new parameters in a corresponding header and has no parameters from a previous header, a new media network communication associated with a new media message sub media stream where the new media network communication has new parameters in a corresponding header and has at least one other parameter from a previous header, a new time network communication associated with the new media message sub media streams where the new time network communication has at least one new parameter in a corresponding header and uses other parameters from a previous header, and a continuation network communication having no new parameters in a corresponding header and has parameters from a previous header. Dividing can include maintaining a user-definable maximum payload size for the network communications. Assigning can include assigning a unique network communication identification parameter for associating the network communication with its sub media stream, and the unique network communication identification parameter can be smaller than a label associated with the sub media stream. Parameters can include one or more of a stream ID parameter, a length parameter, a media type parameter, and a time delta parameter. Queuing the network communications can include according to a media type of the network communication, where the media type includes is one of an audio format network communication, a video format network communication, and a data format network communication. The scheduling process can interleave the audio format network communications, the video format network communications, and the data format network communications according to an interleaving process. The scheduling process can include determining a sub media stream having a lowest ratio, checking the queue of the network communications for a network communication associated with the determined sub media stream, and transmitting the network communication associated with the determined sub media stream when found in the queue.

Another aspect of the subject matter described in this specification can be embodied in a method that includes identifying a location in a pre-defined portion of a network communication to be sent in a client-server environment, wherein the pre-defined portion of the network communication is reserved for random data, inserting cryptographic information into the pre-defined portion of the network communication at the location, and sending the network communication in the client-server environment to facilitate modifying interactions in the client-server environment based at least in part on a result of processing of the cryptographic information. In addition, another aspect of the described subject matter can be embodied in a method that includes receiving cryptographic information inserted into a pre -defined portion of a network communication in a client-server environment, wherein the pre-defined portion of the network communication is reserved for random data, identifying a location of the cryptographic information in the pre-defined portion of the network communication, processing the cryptographic information, and modifying interactions in the client-server environment based at least in part on a result of the processing of the cryptographic information. Other embodiments of these aspects include corresponding systems, apparatus, and computer program products.

These and other embodiments can optionally include one or more of the following features. Identifying the location can include using at least part of the network communication to determine the location. The pre-defined portion can include the random data, and using at least part of the network communication to determine the location can include retrieving a portion of the random data, and determining an index into the pre-defined portion of the network communication based on the retrieved portion of the random data. Moreover, using at least part of the network communication to determine the location can include retrieving multiple different portions of the random data and determining multiple different indices into the pre-defined portion of the network communication based on the different portions of the random data.

Processing the cryptographic information can include establishing a cryptographic key, and modifying interactions in the client-server environment can include initiating an encrypted session using the cryptographic key. In addition, processing the cryptographic information can include authenticating the network communication, and modifying interactions in the client-server environment can include taming on or off a feature of a program operating in the client-server environment.

Particular embodiments of the subject matter described in this specification can be implemented to realize one or more of the following advantages. Communication sessions can be established in which information providers (e.g., media stream distributors) are better able to ensure that consumers of the information comply with the terms of the information transfer. For example, advertising-based providers can advantageously assure that client applications used to access communication sessions include required advertising.

In addition, network communications corresponding to a communication session can be resistant to snooping by those not directly involved in the session. For example, a client can be a software application running in a browser while a remote server can communicate with the client over a network. Embodiments can avoid a user of the client or anyone intercepting the session traffic on the network from being able to interpret or use the communications without using the specific client to which the communication is directed.

In addition, embodiments can advantageously authenticate communication session participants. For example, so that both the client and server know the identity of the other. Knowing the identity can include assurance that a particular person is using the client or that a specific client or client type is being used. Also, the method can allow a server to detect that a given client is legitimate. Being legitimate can include adhering to communication session protocols or being an authorized client implementation.

In addition, embodiments can ensure that when network communications are altered during transmission, the alteration is detected. For example, a server can send a network communication to a client. An intercepting party can alter the network communication (e.g., inserting spam) and then pass the communication to the client. Upon receipt, the client can detect the alteration, thus avoiding undesirable exposure to spam, phishing attacks, viruses, or other malicious or benign alterations.

In addition, embodiments can overcome the often considerable complexities of balancing the demanding nature of streaming media with security, system administration and computer performance issues. Embodiments can allow each area to work together to permit a feasible, scalable and practical implementation. Embodiments can also advantageously provide backwards compatibility with existing non-encrypted, non-authenticated session establishment protocols.

Encrypted sessions can be established using a method that reduces the risk of reverse engineering. Credentials can be established using the method, which credentials can be used to start an encrypted session, or to verify various other pieces of information. The method can resist reverse engineering since the credentials can be, in essence, hidden in plain sight within data that has been used for bandwidth detection. Moreover, the data (with cryptographic information hidden therein) can still be used for other purposes, such as for bandwidth detection or for holding other information.

Other embodiments can be implemented to realize advantages from breaking media streams into smaller network communications such as smooth delivery of each of the sub media streams and smooth playback of the entire stream (e.g., allowing small but more time-sensitive audio messages to be interleaved with much larger but less time-sensitive video messages so that the overall stream can be played back in a pleasing manner).

The details of one or more embodiments of the invention are set forth in the accompanying drawings and the description below. Other features, aspects, and advantages of the invention will become apparent from the description, the drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows an example network environment in which real-time, priority-based communications are imparted over an encrypted communication session.

FIG. **2** shows an example process of imparting real-time priority-based network communications in an encrypted communication session.

FIG. **3**A shows an example network communication.

FIG. **3**B shows an example network communication.

FIG. **4** shows an example network communication control byte.

FIG. **5**A shows an example network communication stream having network communications.

FIGS. **5**B-**5**E show an example state machine receiving network communications.

FIG. **6** shows an example process for scheduling real-time priority-based network communications in an encrypted communication session.

FIG. **7** shows an example process of imparting cryptographic information for use in establishing encrypted sessions and enabling/disabling features in a media player/server system.

FIG. **8** shows another example network environment in which real-time, priority-based communications are imparted over an encrypted communication session for use in playing media content.

FIG. **9** represents an example sequence of interleaved audio and video network communications in a real-time priority-based communication system.

FIG. **10** shows an example process for establishing an encrypted communication session.

Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

FIG. **1** shows an example network environment **100** in which real-time, priority-based communications are imparted over an encrypted communication session. A communication session includes a series of related network communications, which may or may not be encrypted, transmitted over a network (e.g., the Internet). The communication session can be earned over a previously established network connection. For example, a TCP connection cars be established, followed by a handshake **150** to establish an encrypted communication session. The communication session can also be an RTCP or RTP based session or other form of connection-based or connectionless network communication.

The network environment **100** can support both encrypted and non-encrypted sessions, both of which can include the use of a handshake **150**, which can include authentication and other communication protocols. Cryptographic information can be included in a previously existing section of the handshake **150** known to contain random bytes, allowing the cryptographic information to remain hidden in plain sight since the cryptographic information appears random itself (due to the nature of the cryptographic information). Re-using the random byte section in this way can handicap reverse engineering attempts and provide interoperability with previously written software.

The network environment **100** includes a server computer **110**. For example, the server computer **110** can be the FLASH® Media Server provided by Adobe Systems Incorporated of San Jose, Calif. The server computer **110** can use a network **120** (e.g., the Internet) to communicate with one or more first client computers **130** and one or more second client computers **140**. The handshakes **150***a* and **150***b* precede the sessions **135** and **145** and can include cryptographic information from the server **110** that the client(s) **130** might not know is there.

The session startup handshakes **150***a* and **150***b* can include one or more bytes that denote the version of the protocol (e.g., the RTMP protocol) and other information used to open a socket and establish a session. The handshakes **150***a* and **150***b* can also include other information, such as the current time as reported by the operating system (e.g., the number of milliseconds since the system was launched). The handshakes **150***a* and **150***b* include a block of bytes that contain random data, and the block of bytes can also include information useable for authenticating the network communications between server and client, and information for establishing an encrypted session. Such cryptographic information can be sent by the server computer **110** and the client computer(s) **140**, but the client computer(s) **130** need not know that such information is present or be able to send such information.

The first client computer **130** can be a computer programmed to establish non-encrypted sessions with the server computer **110** using the session startup handshake **150***a*. For example, the first client computer **130** can include an older version of the FLASH® Player program, which starts a session with the server computer (e.g., the FLASH® Media Server). The server **110** can include cryptographic information in the block of random data in a network communication forming part of the handshake **150***a*. However, if the first client computer **130** fails to recognize this cryptographic information, and thus doesn't send appropriate return cryptographic information, the session **135** that is established with the server **110** is a non-encrypted session. Note that other aspects of the interactions between the server **110** and the client **130** can also be affected by this failure on the part of the computer **130**, since the server **110** knows after the handshake **150***a* that the computer **130** is running a legacy program that is not aware of the new cryptographic information portion of the communication protocol.

In contrast, the second client computer **140** can be a computer programmed to establish encrypted sessions with the server computer **110** using the session startup handshake **150***b*. For example, the second client computer **140** can include a newer version of the FLASH® Player program, which starts an encrypted session with the server computer (e.g., the FLASH® Media Server). The server **110** can include cryptographic information in the block of random data in a network communication forming part of the handshake **150***b*. The second client computer **140** can recognize this cryptographic information and send appropriate return cryptographic information. This then allows the session **143** that is established with the server **110** to be an encrypted session. Note that other aspects of the interactions between the server **110** and the client **140** can also be affected by the handshake **150***b*, since the server **110** knows after the handshake **150***b* that the computer **140** is running a newer program that is aware of the new cryptographic information portion of the communication protocol. Thus, the cryptographic information used to establish encrypted sessions with new client programs can be added to an existing communication protocol used to establish non-encrypted sessions with old client programs, without the old client programs being aware of the newly added cryptographic information.

FIG. **3**A shows an example network communication **300**. The network communication **300** can be used, for example, as part of the handshakes **150***a* and **150***b* that precedes the sessions **135** and **145**. The network communication **300** can include a time portion **330**, which can include a timestamp or other time information. The time information can be used to determine whether a network communication, such as communication **300**, has been delayed in the network transmission or whether the originating computer has an appropriately set time. A handshake, such as handshake **150***a*, can fail if the time included in time portion **330** is not properly set. Information from the time portion **330** can also be used in cryptographic algorithms relating to network communication **300**.

The network communication **300** can also include a cryptographic information offset **312**. The offset **312** can be used in whole or part to locate cryptographic information **314** in a pre-defined portion **310** that includes cryptographically pseudo-random data. The cryptographic information **314** can appear random and can be indistinguishable from the other random data. In this way, the cryptographic information can be hidden in plain sight. Using this technique can reduce the likelihood that reverse engineering is able to discover the details of the communication protocol.

the location of cryptographic information **314** within the pre-defined portion **310** can vary with implementation or within a given implementation. For example, the cryptographic information can be located at a pre-determined byte location, or a pre-determined byte location (e.g., offset **312**) can contain a value from which the location of cryptographic information **314** can be determined. In some implementations, using cryptographic information can include the use of an encryption key establishment protocol, such as a Diffie-Hellman key exchange or other suitable technique for establishing one or more symmetric or asymmetric encryption keys.

In some implementations, using cryptographic information can include the use of authentication information **324**. Authentication information **324** can be a type of message authentication code (MAC) and can be used to simultaneously verify both the data integrity (e.g., the data was not altered in transit) and the authenticity (e.g., the originator of the information is who it claims to be) of a network communication received over a communication session. Authentication information **324** can be calculated using a specific algorithm involving a cryptographic hash function in combination with a secret key, such as a hash message authentication code (HMAC). Any iterative cryptographic hash function, such as MD5, SHA-1, or SHA-256 can be used in the calculation of an HMAC. The cryptographic strength of the HMAC depends upon the cryptographic strength of the underlying hash function, on the size and quality of the key and the size of the hash output length in bits.

The location of authentication information **324** cart be found similarly to the cryptographic information **314**. For example, authentication information offset **322** can be used in whole or part to locate authentication information **324** within a pre-defined region **320** containing cryptographically pseudo-random data. The authentication information **324** can also be located at a pre-determined byte location, or a pre-determined byte location can contain a value (e.g., offset **322**) from which the location of authentication information **324** can be determined.

In addition, although the locations of cryptographic information **320** and authentication information **324** are shown in FIG. **3**A as being entirely contained within the pre-defined portions **310** and **320**, respectively, it will be appreciated that either the beginning or end of the cryptographic information can be at the beginning or end of the pre -defined portion **310** or **320**. For example, cryptographic information **314** can be adjacent to the cryptographic information offset **312** (if one is present, or the time portion **330**, If one is not present) or the authentication information offset **322** or predefined portion **320**.

FIG. **3**B shows an example network communication **350**. A network communication is a part of a media stream that has been broken down in stream processing. Network communication **350** includes network communication control byte **360**, header **390** and data payload **370**. Network communication control byte **360** can contain state information relating to the purpose of network communication **350** and information allowing network communication **350** to be matched with a set of stored values of parameters. Header **390** can contain new values of parameters corresponding to previously stored parameter values. Network communication **350** can also not include a header **390** or include an empty header **390**, and thus not include any new values of parameters. Data payload **370** can be a fixed length data segment that is part of a media stream (e.g., audio, video or data accompanying a video stream). In order to be played, for example, the data payload **370** can be decrypted using the cryptographic information **314** (or other cryptographic information derived there from) and reassembled in the proper order with other payloads that are part of the same media stream.

In some embodiments, version byte **380** can be used to provide information on the particular version of the transport protocol being used. Version byte **380** can be used in conjunction with a handshake (e.g., handshakes **150***a* and **150***b*) and a network communication **300**, including cryptographic information, or a network communication **350**, including data relating to media stream content. The inclusion of this information can allow easier maintenance of future protocol versions and backwards compatibility. In operation, only a single version byte would likely be necessary per session. Therefore, version byte **380** would likely not add any overhead on a per network communication basis.

In attempting to reduce the overhead of stream messages, certain assumptions can be leveraged into bandwidth savings. One such assumption involves the necessity of a 32-bit message stream ID. Statistically, there are usually less than sixty-one (61) message streams (having unique message stream IDs) in a server at any given time for any given client. Sixty-one (61) message streams can be uniquely represented in only 6 bits. Thus, having a 32-bit message stream ID is typically a waste of space.

FIG. **4** shows air example network communication control byte **400**. Network communication control byte **400** can be the same as network communication control byte **360**. Network communication control byte **400** includes a network communication identifier, such as sub media stream ID **410** and network communication type **420**. While the statistics suggest that there are a maximum of sixty-one (61) message streams in a server at any one time for a particular client, artificially limiting the maximum number of streams is not an efficient means for obtaining bandwidth at the expense of stream capacity. Sub media stream ID **410** instead provides a mechanism for supporting 65,597 total simultaneous streams (using a range of 3 to 65,599). Sub media stream ID **410** includes bits 0-5 of network communication control byte **400**. As provided in Table 1, when the value of sub media stream ID **410** lies between 3 and 63 in some embodiments, sub media stream ID **410** is assigned at face value. Statistically, this situation will occur most often, resulting in a 6-bit sub media stream ID. However, when the value of sub media stream ID **410** is 0 in some embodiments, a second byte is added to sub media scream ID **410** that represents the range of 54 to 319. When the value of sub media stream ID is 1 in some embodiments, two additional bytes, a second byte and a third byte, are added to sub media stream ID **410** that represent the range of 320 to 65,599. Therefore, the embodiments represented by FIGS. **3**B and **4** support tip to 65,597 streams, while still conserving bandwidth on a statistically large number of messages. A sub media stream ID **410** value of 2 in some embodiments represents that the network communication contains a low-level protocol message, which is used to control administrative aspects of the embodiments represented by FIGS. **3**B and **4**.

TABLE 1

Sub media stream ID (Bits 0-5 of Control Byte)	
Value	Meaning
3-63	Sub media stream ID is the face value of 3-63.
0	There is another byte that represents the range 64-319 (Second byte + 64).
1	There are two more bytes representing the range 320-65,599 ((Third byte)*256 + (Second byte + 64)).
2	The network communication is a Low-Level Protocol Message.

The configuration of the real-time priority-based protocol represented by FIGS. **3**B and **4** advantageously leverages the statistical maximum of sixty-one (61) streams. Because there are usually less than sixty-one (61) streams in the server at any given time for a client, only the 6 bits of sub media stream ID **410** will generally be sent. Thus, the 32-bit header for message stream IDs of the current technologies can be reduced to only 6 bits for the majority of data streams. In rare cases, in which the number of streams exceeds 61, the additional bytes of information can be included to support those extra streams.

The configuration of sub media stream ID **410** in some embodiments, as provided in Table 1, demonstrates implementation of a state machine having different state associations using its first 2 bits to address many different levels of streams. In furtherance of the state machine concept implementing the embodiments, network communication type **420** provides four states/types for classifying each network communication message. Network communication type **420** identifies the type or purpose of the network communication.

As provided in Table 2, bits **6** and **7** of network communication control byte **400** includes network communication type **420**. When the value of network communication type **420** is 0 in some embodiments, the network communication state or type indicator is a new/reset message type (Type 0) or indicates a new stream. This signifies the first message sent on the network communication stream in some embodiments. As is described in Table 2, a Type 0 message can include additional bytes of information that include the typical header information of the current stream methods, such as message length, message type ID, and message stream ID. Instead of using a timestamp, however, the embodiments represented by FIGS. **3**B and **4** use a time delta (A), or the change in time from the previous message. The time-delta (A) is essentially an absolute time because it is usually added to ‘0’ time. Because the Type 0 network communication includes a considerable amount of information, it will generally be larger than the subsequent network communications in the stream.

When the value of network communication type **420** is 1 in some embodiments, the network communication state or type is a new media indicator of a specified new time change/time delta, length, and any new message type ID (Type 1). This state can signify a new network communication that is measured using the new specified time delta (i.e., the time stamp change from the last message) and the specific length. Thus, a subsequent message in a network communication stream that belongs to the same sub media stream ID, but that is a different type/format and a different time delta than the previous network communication only has to include header information on the message type ID, time delta, and the length. A Type 1 network communication can, therefore, have fewer bytes of header information than the Type 0 network communication, because the state machine implementing the represented embodiments has been set with certain information by the Type 0 network communication that remains unchanged for the Type 1 network communication.

A value for network communication type **420** of 2 in some embodiments represents the network communication state or type for a new message defined only by a single piece of header information, such as the time delta (Type 2) in a new time indicator. A Type 2 network communication can have all of the same information from the previous network communication except possibly for a change in the time delta. Therefore, the state machine might only need to use the new time delta to change to the proper new state.

The least amount of overhead or header information in the embodiments represented by FIGS. **3**B and **4** accompanies the Type 3 network communication, when the value of network communication type **420** is 3. A Type 3 network communication is a continuation indicator. A continuation indicator network communication can use the same message stream ID, the same message type ID, the same time change/time delta, and the same length of the previous network communications. Therefore, the state machine implementing this embodiment can be already set to the appropriate state. One design goal can be to optimize the streaming process to get the network communications to Type 3 network communications as soon as possible. Therefore, Type 3 network communications that are essentially continuations of a same message are transmitted with a 0-length header.

The 0-length header of the Type 3 network communications is especially interesting when the Type 3 network communication represents the begriming of a new message. This scenario can occur when two consecutive network communications use the same message stream ID, the same message type ID, the same time change/time delta, and the same length as the previous network communication. In this case, a new message is enabled without any required header information, a 0-length header.

TABLE 2

Network Communication Type (Bits 6-7 of Control Byte)

Bits	Description
00	(Type 0) Begin new message. Reset all information. Type 0 is used when the time is restarted or for the first message sent on the network communication stream. A Type 0 network communication message resets the time to 0 before adding the supplied time delta (A). A Type 0 network communication message can contain the following header information: 3 bytes - message time delta (A) 3 bytes - message length 1 byte - message type ID 4 bytes - message stream ID
01	(Type 1) Begin new message using specified A, new length, and/or new message type ID. A Type 1 network communication message can contain the following header information: 3 bytes - A 3 bytes - new message length 1 byte - message type ID
10	(Type 2) Begin new message using specified A. A Type 2 network communication message can contain the following header information: 3 bytes - A
11	(Type 3) Continuation (possibly start new message). A Type 3 network communication message can contain no header information, containing just the raw data/payload.

FIG. **2** shows an example process **200** of imparting real-time priority-based network communications in an encrypted communication session. A communication session for a media stream is established **210** using cryptographic information in a pre-defined portion of a network communication. For example, the cryptographic information in portion **310** can be used by the computer **140** to establish an encrypted session **145** with server computer **110**. Also, authentication information in portion **320** can be used to verify the integrity and authenticity of network communications in the communication session.

Values of parameters relating to a sub media stream, included in a header of a first network communication, are received **220**. For example, the parameters (e.g., a stream ID, media stream length, media type and time delta) can be included in header **390** of network communication **350**. The value of the parameters are stored **230**. For example, software on computer **140** can store, in physical memory or on a hard drive, the values of parameters across multiple network communications.

State information in a control portion and a data payload included in a second network communication is obtained **240**. For example, the state information can be included in control byte **360** and the data payload can be payload **370**. The state information can include network communication type **420** and sub media stream ID **410**, allowing, in some embodiments, the purpose and sub media stream association of the second network communication to be obtained in 1 byte of information (e.g., control byte **360**).

The purpose of the second network communication is identified **250** from the state information. For example, network communication type **420** can indicate that the network communication is starting a new sub media stream or that a new media type is being transmitted over the sub media stream (see Table 1). Process **200** determines **260** whether the header of the second network communication includes new parameter values. For example, some network communications (e.g., Type 3) can include no new parameter values (or a header) while others (e.g., Type **1**) can include multiple new parameter values. When the header of the second network communication includes new parameter values, the state information can indicate that the stored values are to be updated **280** with the new values. Process **200** can look for new parameter values in response to the purpose of the network communication. For example, there can be times when network communication will include extraneous header information, such as values for parameters that should not be used to update the stored values. The purpose can enable process **200** to avoid incorrectly updating stored parameter values.

The data payload is processed **270** using the stored parameter values and the identified purpose. For example, payload **370** can be correctly assembled and played based on the stored values of time deltas and media type indications. The data payload can also be process as the beginning network communication in a new sub media stream based on that being identified as the purpose.

FIG. **5**A shows an example network communication stream **500** having network communications **502**-**508**. Network communication **502** is a Type 0 network communication with a sub media stream ID of 4. Because network communication **502** is a Type 0 network communication, it includes additional header data **512**, making network communication **502** relatively larger in terms of total bytes **514** than network communications **504**-**508**. The additional data, includes the message stream ID (12346), the message type ID (9), the message length (435), and the time delta (20). This header information is set into the state machine. Network communication **504** is a Type 3 network communication belonging to the same sub media stream ID 4. Because network communication **504** belongs to the same media stream ID and is the same type and length as network communication **502**, the state set by the extra header information of network communication **502** already defines how to handle network communication **504**. According to some embodiments, network communication **504** only needs to designate the sub media stream ID and network communication type. Therefore, only a single byte of overhead is added to the data payload of network communication **504**, as represented by the total bytes **514**. Network communications **506** and **508** are also Type 3 continuation network communications that require only an additional byte of control information to adequately handle those network communications.

FIGS. **5**B-**5**E show an example state machine **510** receiving network communications **502-508**. FIG. **5**B shows state machine **510** receiving a Type 0 network communication. State machine **510** sees that network communication **502** is a Type 0 network communication and expects each of the state variables to be set with the header information contained in network communication **502**. In some embodiments, state machine **510** can also include message count **516** that keeps track of the progress of each receiving message. Network communication **502** brings 128 bytes of data payload. Therefore, using the message length parameter, state machine **510** can subtract 128 bytes from the total length, 435 bytes, to hold the remaining message expected length of 307 bytes in message **516**. The state machine represented by FIG. **5**B is used in both the send direction and the receive direction. As network communications arrive at the receiver system, the state machine is used in reverse to translate the state assignments necessary to play the data stream network communication. Therefore, the state machine for the real-time priority-based communication system can be located at least at the sending entity and the receiving entity.

FIGS. **5**C-**5**E show state machine **510** receiving Type 3 network communications. In FIG. **5**C, state machine **510** reads that network communication **504** is a Type 3 network communication, and therefore, expects that only the network communication type and sub media stream ID state variables will be changed. Because network communication **504** could belong to a different sub media stream than that of network communication **502**, the sub media stream ID is used in the Type 3 continuation network communications. The remaining state variables set by network communication **502** remain the same. Message **516** is changed again to reflect the 128 bytes of payload data delivered by network communication **504**, 179 bytes now remain in the message. In FIG. **5**D, state machine **510** reads that network communication **506** is also a Type 3 network commumcation, and therefore, also expects that only the network communication type and sub media stream ID state variables will be changed. The remaining state variables set by network communication **502** continue to remain the same. Message **516** is again changed to reflect the 128 bytes of payload data delivered by network communication **504**. 51 bytes now remain in the message. FIG. **5**E also represents state machine **510** receiving a Type 3 continuation network communication. With this receipt, because state machine **510** knows the maximum network communication data payload length is 128, state machine **510** knows that this will be the last network communication for this particular message, as reflected in message **516** zeroing out. The next network communication will, by process of elimination, have to be for a new message (or be a low-level protocol message).

Referring back to Table 1, when the value of sub media stream ID 40 is 2, in some embodiments, the state is designed to change for the system to expect a low-level protocol message. Table 3 describes two low-level protocol messages that can be implemented in various embodiments. In such embodiments, a message ID of 1 represents a change to the maximum network communication size. The parameter passed with such a message ID can be an integer representing the size for all future network communications. Another message ID, shown as message ID 2 in Table 3, can allow for an abort signal to be sent to the network corresponding to a particular sub media stream ID. For example,

if a communication session is in mid-stream when the transmitting end state-down, or other such event, the abort message would generally inform the network that it should not expect any further network communications, thus, aborting the stream. Many other optional low-level protocol messages can also be provided for in various embodiments. The two messages provided for in Table 3 are merely examples.

TABLE 3

Low-Level Protocol Messages

Message	Message ID	Message Data	Description
Set Network Communication Size	1	4 bytes representing an integer. Bytes are sent in "network" order (i.e., order of significance)	The integer represents the new maximum network communication size for all future communications.
Abort Message	2	4 bytes representing an integer. Sent in "network" order.	The integer represents a sub media stream ID. If that network communication stream is currently waiting to complete a message (i.e., expecting more communications). The message stream is aborted.

Some embodiments obtain benefits from breaking the individual messages further down into network communications. While it might appear simpler to send one message and then the next, a linear process, such as this, can be much loss efficient. A stream representing some media presentation can include several different sub media streams. One sub media stream can be for the audio, one for the video, and another for data. The combination of each of these sub media streams makes up the entire streaming presentation. Therefore, it becomes important to coordinate smooth delivery of each of the sub media streams to facilitate the smooth playback of the entire stream. For example, a video key frame can have a large amount of data in contrast to audio messages that are generally very small. However, the audio messages are much more time sensitive than the video messages due to the nature of human hearing. Therefore, if an audio message is sent from the audio sub media stream, followed by a very large video message from the video sub media stream, followed by the next audio message, the audio stream can typically stall because the next audio message simply might not arrive in time.

In operation, the larger video messages are typically much less frequent, even though they can be much larger. Therefore, interleaving other messages in between the video messages is acceptable as long as the video message eventually arrives in time for the smooth playback of the video sub media stream. Considering this balancing process, breaking the messages into network communications that can be more easily balanced and interleaved increases the efficiency of the streaming. While no one maximum network communication size is best for all situations, a given maximum can be beneficially set in which the video and data messages arrive without too much effect on the timely arrival of the audio network communications.

In some embodiments, the maximum network communication size is set to correspond to the complete length of a standard audio message in the audio sub media streams in order to ensure timely delivery of the complete audio message. For example, the sampling rate for the audio compression algorithm, Nellymoser, can be 8 kHz. The 8 kHz sampling typically generates messages that are 128 bytes long. Therefore, in some embodiments that can use the Nellymoser compression algorithm, the maximum network communication size can be set to 128 bytes. Other audio compression formats can produce messages that are either greater than or less than 128 bytes. In such cases, a low-level protocol message can be used to change the maximum network communication size of the applicable embodiments to achieve optimal performance.

FIG. **6** shows an example process **600** for scheduling real-time priority-based network communications in an encrypted communication session. Balancing the transmission of the network communications in the different media sub media streams can be controlled by a scheduling process that directs which network communication of which stream or sub media stream to send next. Each type of media can be assigned a specific priority value. For example, because audio messages are the most sensitive, they can be assigned the highest priority value. Video and data messages would each likely be assigned lower priority values.

The priority of each sub media stream (Prioritys_B) is summed **602** into a total stream priority (StreamPriority). A desired proportion for sending each sub media stream within the stream is calculated **604** according to formula:

$$\text{Desired Proportion}=\text{Priority}_{SB}/\text{StreamPriority}$$

The total number of network communications sent (Count) are counted **606** per sub media stream over a group of preceding network communications (N). N represents a scheduling sampling window that can be pre-set by a user or developer. The sampling window can be based, on a period of time or a count of network communications. For example, process **600** can count network communications sent (Count) over a period of one second or 5 seconds. The network communications for each sub media stream can be counted **606** concurrently or sequentially. Process **600** can also send a pre-set number of network communications and count **606** the number of network communications seat in each sub media stream out of the pre-set number. The actual proportion of network communications sent per sub media stream in the preceding N network communications is developed **608** according to formula:

$$\text{Actual Proportion}-\text{Count}/N$$

A ratio of desired proportion to actual proportion is determined **610** for each sub media stream. The sub media stream with lowest ratio is selected **612**. The process **600** determines **614** whether the selected sub media stream has a network communication queued. If a network communication is queued for the selected sub media stream, the sub media stream network communication is sent **616**. If no data is queued for the selected sub media stream, the determination **618** is made whether there are any other waiting sub media streams left. If there are no remaining sub media streams, then no network communications are sent in the cycle **620**. However, if there are remaining sub media streams, a new sub media stream is selected **622** that has the next lowest ratio of desired to actual proportions. The process **600** can repeat the operations **614-622**, depending on the case, until all network communications in each sub media stream are sent.

Scheduling process **600** can result in a favorable balance of streams. If all sub media steams always have data to send, not only does scheduling process **600** provide for those streams to obtain the proper proportion of network communications sent, but also provides a good interleaving mix over the scheduling window. When some sub media streams do not have data or, at least, do not have data queued, scheduling process **600** allows the tower priority sub media streams to use a greater bandwidth. Thus, bandwidth use is routinely optimized. One embodiment can achieve desired results with audio having a priority of 100, data priority of 10, and video priority of 1. However, other various priority assignments can prove better for different situations and known or expected data types or data loads.

FIG. **7** shows an example process **700** of imparting cryptographic information for use in establishing encrypted sessions and enabling/disabling features in a media player/server system, such as the client-server environment **800**, FIG. **8**. The process **700** includes operations for authentication, determining encryption parameters, and turning on or off features associated with the network communication. For example, the network communication can involve a user employing a FLASH® Player program to play media available from the FLASH® Media Server program. In some implementations, features cm be enabled or disabled based on version information that can be separate from the cryptographic information. For example, feature availability (e.g., audio or visual capabilities) might depend on the version of the FLASH® Player program installed on the user's client computer.

A first portion of the random data can be retrieved **702**. For example, in the network communication **300** sent by client computer, the server can look in a pre-determined byte position within the pre-defined portion **310**. A first index into the pre-defined portion can be determined **704** based on the retrieved first portion. For example, one or more bytes of the random data can be used as the dividend in a modulo operation, where the divisor is the length of the region of the pre-defined portion **310** set aside for a message authentication code (e.g., an HMAC), minus the length of the message authentication code. The first index can then be set equal to the remainder of this modulo operation plus a pre-defined offset (which can be zero).

Note that various combinations of the random data can be used to generate the dividend. For example, x bytes of the random data can be treated as a single binary number forming the dividend failing in the range of zero to $^{2(8}1$, or the same x bytes of the random data can be treated as x binary numbers that are added together to form the dividend falling in the range of zero to $x(2^8-1)$. Various other combinations of the random data are also possible. In addition, the first index can be determined from the first portion of the random data alone, or from the first portion in combination with other information retrieved from the network communication. For example, such other information can come from the header **330** or the payload **340**.

In any event, once the index is determined, this index corresponds to the starting position of the cryptographic information used for message authentication (e.g., the starting point of the HMAC) in the block of otherwise random data. The network communication is authenticated **706** rising this cryptographic information. For example, the authentication can involve using the first index to access an HMAC in the network communication **300**. If the message authentication code is not confirmed **708**, then the process **700** can check **710** whether a fallback first portion is available. This occurs when the first index determined **704** fails to locate a message authentication code usable to authenticate **706** the network communication.

In this case, one or more fallback processes can be provided for retrieving **702** the first portion and determining **704** the first index. The fallback processes can use a different technique for retrieving **702** the first portion and/or determining **704** the first index, determining **704** the first index. These fallback processes can provide additional security for the authentication process, allowing a server system to change the indexing technique when a currently used technique has been discovered, and the client computers can then automatically fall back to the new indexing technique. Once all available fallback processes have been tried, the process **700** ends without the network communication having been authenticated.

If the message authentication code is confirmed **708**, then a second portion of the random data is retrieved **712**. For example, in the network communication **300** sent by client computer, the server can look in another pre-determined byte position within the pre-defined portion **310**. A second index into the pre-defined portion can be determined **714** based on the retrieved second portion. For example, one or more bytes of the random data can be used as the dividend in another modulo operation, where the divisor is the length of the region of the pre-defined portion **310** set aside for encryption parameters (e.g., Diffie Hellman information), minus the length of the encryption parameters. The second index can then be set equal to the remainder of this modulo operation, plus a pre-defined offset (which can be zero).

As with the message authentication code, various combinations of the random data can be used to generate the dividend. For example, x bytes of the random data can be treated as a single binary number forming the dividend falling in the range of zero to $2^{(8x)}-1$, or the same x bytes of the random data can be treated as x binary numbers that are added together to form the dividend falling in the range of zero to $x(2^8-1)$. Various other combinations of the random data are also possible. In addition, the second index can be determined from the second portion of the random data alone, or from the second portion in combination with other information retrieved from the network communication. For example, such other information can come from the header **330** or the payload **340**. Moreover, it will be appreciated that the random data section needs to be larger than the total length of the message authentication code and the encryption parameters, e.g., larger than one hundred and sixty bytes when using a thirty two byte HMAC and one hundred and twenty eight bytes of Diffie Hellman information.

In any event, once the index is determined, this index corresponds to the starting position of the encryption parameters used for establishing a cryptographic key (e.g., the starting point of the Diffie Hellman information) in the block of otherwise random data. This cryptographic information in the network communication is confirmed **716**. This confirmation can involve using the second index to access and confirm encryption parameters in the network communication **300**. If the encryption parameters are not confirmed **718**, then the process **700** can check **720** whether a fallback second portion is available. This occurs when the second index determined **714** fails to locate encryption parameters usable to initiate encrypted communications in the network environment.

In this case, one or more fallback processes can be provided for retrieving **712** the second portion and determining **714** the second index. The fallback processes can use a different technique for retrieving **712** the first portion and/or determining **714** the second index. Note that these processes can also be different than those used for the first index. These fallback processes can provide additional security for the encrypted session establishment process, allowing a server system to change the indexing technique when a correctly used technique has been discovered, and the

client computers can then automatically fall back to the new indexing technique. Once all available fallback processes have been tried, the process **700** ends without the encryption parameters having been confirmed.

If the encryption parameters are confirmed **718**, then a determination **722** can be made if an encrypted session is desired. For example, the encryption parameters can contain information that the session (e.g., session **145**) is to be an encrypted session. If so, then the encrypted session is initiated **724**. Otherwise, an un-encrypted session can be initiated. Either session (e.g., encrypted or un-encrypted) can be between the client computer **802** (e.g., executing FLASH® Player program) and the media server **804** (e.g., FLASH® Media Server), in FIG. **8**.

If it is determined **726** that one or more features are to be enabled or disabled, then one or more features of the media player, the media server or both, are turned on or off **728**. For example, a set of features can be turned on or off according to tire version number of the client media player. This version number can be provided by the client media player (e.g., in non-encrypted and non-disguised form in the handshake **150**), and the server can decide to trust the version number provided based on the fact that the client properly incorporated cryptographic information within the random byte section of a network communication (e.g., the client media player is not a legacy player that has been modified to improperly identify itself as a newer version).

FIG. **8** shows another example network environment **800** in which real-time, priority-based communications are imparted over an encrypted communication session for use in playing media content. The network environment **800** includes a client computer **802** and a media server **804**. The media server **804** can provide media content **806** to the client computer **802**. For example, media server **804** can include a FLASH® Media Server program. The media content **806** can include web applications, games and movies, and multimedia content for client computers (e.g., home personal computers, mobile phones, personal digital assistants, smart phones, or various embedded devices.)

The client computer **802** can include software, firmware and hardware. The hardware can include a computer readable medium **812**, a processor **814**, and one or more interface devices **816**. The computer readable medium **812** can include one or more hard drives, external drives, magnetic disks, optical disks, tape drives, memories devices, etc. The processor **814** can include one or more central processing units capable of interpreting computer program instructions and processing data, and the processing units can include one or more processor cores. The interface devices **816** can include one or more display and audio devices (e.g., computer screens, computer monitors, digital displays, liquid crystal displays (LCDs), light emitting diodes (LEDs), etc.) and audio-capable components (e.g., microphones, speakers, etc.). The interface devices **816** can support a graphical user interface (GUI) by which the user sees, hears and experiences the output of a media player application **808**.

The software/firmware can include the media player application **808** and an application execution environment **810**. For example, the media player application **808** can be a FLASH® Player program installed on a home computer or other electronic device. The media player application **808** can run in the application execution environment **810**, which can be an operating system (OS) for the computer **802**, or a cross-OS runtime environment installed on the computer **802**, such as the Adobe® AIR™ runtime environment available from Adobe System Incorporated of San Jose, Calif.

The random byte section that embeds the cryptographic information can be included in the network communications between the client computer **802** and the media server **804**. For example, the random byte section cars be generated by a users FLASH® Player program and by the FLASH® Media Server program. As noted above, cryptographic information (e.g., including Diffie Hellman key exchange and HMAC information) can be injected into the random byte section at pre-determined or program-determined locations. In some implementations, the locations can be determined by various processes, which can use pieces of the random data to index the locations of the cryptographic information. The receiving end of the communication, knowing the new protocol, can locate and remove the cryptographic information from the random byte section. If the cryptographic information can be verified, then the receiving side knows that the new protocol is being used. If the cryptographic information cannot be verified, one or more fallback positions can be checked before determining that the new protocol is not being used (because the expected cryptographic information cannot be found in the random byte section), and thus the communication is of a legacy type. Note that from the perspective of an external eavesdropper, the cryptographic information is seen as nothing more than the previously included random data, which can be used for bandwidth detection.

FIG. **9** represents an example sequence of interleaved audio and video network communications in a real-time priority-based communication system **900**. System **900** shows a one-way communication session of streaming media occurring from transmitting entity **920** to receiving player **922** over network path(s) **950**. The messages to be streamed can include audio, video, and data messages. Network path(s) **950** can support an encrypted communication session, which can be transmitted over a network connection. FIG. **9** shows a message that includes video and audio sub media streams. As the video and audio sub media streams are broken into video and audio network communications, the state machine of the various embodiments supply the necessary header information for the network communications. Transmitting entity **920** places the network communications in either video queue **924** or audio queue **926** to await transmission.

The scheduler at transmitting entity **920** controls when network communications are sent to network interface **932** from either video or audio queue **924** and **926**. Network communications **901**, **903**, **905**, **907**, **909**, **911**, **913**, and **915** are represented on network path(s) **950** along with corresponding header parameters **902**, **904**, **906**, **908**, **910**, **912**, **914**, and **916** configured according to the example. Because audio network communications are more sensitive than video network communications, the scheduler sends audio network communication **901** over network path(s) **950** first. Audio network communication **901** includes header **902**. Header **902** describes audio network communication as a Type 0 network communication and gives the necessary information that accompanies a Type 0 new message network communication, including the network communication ID, message stream ID, length, message type, and time delta. The state machine at receiving player **922** sets the state for the audio sub media stream beginning with audio network communication **901**. Audio network communication **903** is the next network communication sent by the scheduler at transmitting entity **920** and includes header **904**. Header **904** identifies network communication **903** as a Type 3 continuation header belonging to network communication ID 1, the network communication ID assigned to the first audio sub media stream. Because audio network communication **903** is a continuation network communication, only the network communication type and network communication ID are required in the header information. The state machine at receiving entity **922** reads that network communication **903** is a Type 3 continuation network communication and keeps all states fixed for the audio sub media stream of network communication ID 1.

The scheduler at transmitting entity **920** sends video network communication **905** next along with header **906**. Because video network communication **905** is the first of the video sub media stream, it is designated a Type 0 network communication and includes all of the necessary header information to go along with the first network communication of the video sub media stream. Thus, video network communication **905** can include a different network communication ID, message stream ID, length, message type, and time delta. The state machine at the receiving entity sets the necessary states for the video sub media stream of network communication ID 2. The next network communication sent by the scheduler is audio network communication **907** and header **908**. Header **908** identifies network communication **907** as a Type 3 continuation network communication of network communication ID 1. The state machine refers to the states set for the audio sub media stream of network communication ID 1 and maintains those settings for the first audio sub media stream.

The scheduler then sends audio network communication **909** and header **910** across network path(s) **950**. Header **910** identifies audio network communication **909** as a Type 0 new message network communication. The state machine at receiving entity **922** resets all information in the states of the audio sub media streams and sets the new states according to header **910**. The next network communication in the interleaved sequence is video network communication **911** along with header **912**. Header **912** identifies video network communication **911** as a Type 3 continuation network communication of video sub media stream network communication ID 2. The state machine at receiving entity **922** accesses the states set for network communication ID 2 and keeps all of the state information in tact for video network communication **911**. The next network communication sent on network path(s) **950** is audio network communication **913** and header **914**. Header **914** identifies network communication **913** as a Type 3 continuation network communication of network communication ID 3. The state machine accesses the states saved for the audio sub media stream of network communication ID 3 and maintains each state intact. The final network communication shown in FIG. **9** is audio network communication **915** and header **916**. Header **916** identifies network communication **915** as a Type 2 network communication representing a new message having a new time delta. The state machine reads that network communication **915** is a Type 2 network communication and expects the beginning of a new audio sub media stream with only a change to one of the states for a new network communication ID 4.

As each of network communications **901**, **903**, **905**, **907**, **909**, **911**, **913**, and **915** are received at receiving entity **922** through network interface **934**, the audio network communications are reassembled in audio assembly **930**, while the video network communications are reassembled in video assembly **928** before being played back at receiving entity **922**. The state machine at receiving entity **922** can read the parameters in each of headers **902**, **904**, **906**, **908**, **910**, **912**,**914**, and **916**, and controls the reassembly of the network communications in audio assembly **930** and video assembly **928**. In additional embodiments, other interleaving scheduling processes and/or efficiency processes can be implemented. The goal is to achieve an efficient mix of audio, video, and data network communications that results in a good playback that minimizes the audio or video jitter in the playback.

FIG. **10** shows an example process **1000** for establishing as encrypted communication session. A responsive handshake network communication, including an outgoing random portion, is generated **1010**, using an incoming random portion from the handshake network communication. For example, the handshake network communication can be network communication **300**. The authentication information **324**, cryptographic information **314**, random data from portions **310** and/or **320**, or a combination of these can be used as a random portion. The random portion can be used to avoid replay attacks since each response is based on unique information (e.g., pseudo-random data) to the handshake network communication. Retransmitting a captured handshake network communication can result in failure to establish an encrypted communication session because the handshake network communication would not be based on a "one-time use" challenge.

The responsive handshake network communication is communicated **1020**. For example, after a handshake network communication, such as network communication **300**, is received, a responsive handshake network communication, similar to network communication **300**, is communicated **1020** to the originator of the handshake network communication as a part of completing the encrypted communication session handshake.

A second network handshake is received **1030** in response to communication the responsive handshake network communication. For example, the second network handshake can include a personalized response based on the outgoing random portion communicated **1020** with the generated **1010** responsive handshake network.

The second handshake network communication is authenticated **1040** based on the outgoing random portion. For example, the other party to the handshake can verify that indeed the handshake communications are associated with the present handshake attempt rather than being intercepted and subsequently retransmitted handshake communications (e.g., in a replay attack). The authentication is possible because the random data in the handshake network communications changes with each handshake communication sent or received and allows unique, one-time use.

Embodiments of the subject matter and the functional operations described in this specification can be implemented in digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Embodiments of the subject matter described in this specification can be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a computer-readable medium for execution by, or to control the operation of, data processing apparatus. The computer-readable medium can be a machine-readable storage device, a machine-readable storage substrate, a memory device, or a combination of one or more of them. The term "data processing apparatus" encompasses all apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code

that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them.

Storing can include any maintenance in volatile or non-volatile memory or storage. For example, in a file system, hard drive, database, buffer, cache, random access memory (RAM), Compact Flash or Secure Digital card, or network repository.

A computer program (also known as a program, software, software application, script, or code) can be written in any form, of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program does not necessarily correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data (e.g., one or more scripts stored in a markup language document), in a single file dedicated to the program in question, or in multiple coordinated files (e.g., files that store one or more modules, sub-programs, or portions of code). A computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a communication network.

The processes and logic flows described in this specification can be performed by one or more programmable processors executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application-specific integrated circuit).

Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read-only memory or a random access memory or both. The essential elements of a computer are a processor for performing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto-optical disks, or optical disks. However, a computer need not have such devices. Moreover, a computer can be embedded in another device, e.g., a mobile telephone, a personal digital assistant (PDA), a mobile audio player, a Global Positioning System (GPS) receiver, to name just a few. Computer-readable media suitable for storing computer program instructions and data include all forms of non-volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto-optical disks; and CD-ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in. special purpose logic circuitry.

To provide for interaction with a user, embodiments of the subject matter described in this specification can be implemented on a computer having a display device, e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor, for displaying information to the user and a keyboard and a pointing device, e.g., a mouse or a trackball, by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input.

Embodiments of the subject matter described in this specification can be implemented in a computing system that includes a back-end component, e.g., as a data server, or that includes a middleware component, e.g., an application server, or that includes a front-end component, e.g., a client computer having a graphical user interface or a Web browser through which a user can interact with an implementation of the subject matter described in this specification, or any combination of one or more such back-end, middleware, or front-end components. The components of the system can be interconnected by any form or medium of digital data communication, e.g., a communication network. Examples of communication networks include a local area network ("LAN") and a wide area network ("WAN"), e.g., the Internet.

The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

While this specification contains many specifics, these should not be construed as limitations on the scope of the invention or of what might be claimed, but rather as descriptions of features specific to particular embodiments of the invention. Certain features that are described in this specification in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features might be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination might be directed to a subcombination or variation of a subcombination.

Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing might be advantageous. Moreover, the separation of various system components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems can generally be integrated together in a single software product or packaged into multiple software products.

Thus, particular embodiments of the invention have been described. Other embodiments are within the scope of the following claims. For example, the actions recited in the claims can be performed in a different order and still achieve desirable results. The random data need not be truly random data, but rather can be pseudo random data. Moreover, the pre-defined portion of the network communication need only be reserved for random data, but need not actually include random data in all implementations.

What is claimed is:

1. A computerized method comprising:

at a server system, establishing, based at least in part on cryptographic information placed in a predefined portion of a handshake network communication, a communication session with a client system for streaming one or more media streams from the server system to the client system;

processing a stream of the one of more media streams, the processing comprising:

generating a first network packet of the stream, the first network packet having a first sub stream identifier (ID) in a control portion of the first network packet; and

generating a second network packet of the stream, the second network packet having a second sub stream ID in a control portion of the second network packet, wherein fewer bits are allocated for representing the second sub stream ID in the control portion of the second network packet than for representing the first sub stream ID in the control portion of the first network packet by storing a first value at bit positions of the control portion in the second network packet that represents the second sub stream ID while storing a second value at the bit positions of the control portion in the first network packet to indicate that additional bit positions are included in the first network packet to represent the first sub stream ID; and

streaming the processed stream to the client system via the established communication session by concatenating the first network packet and the second network packet.

2. The method of claim **1**, wherein the second value is generated to indicate a number of bits used to represent the first sub stream ID.

3. The method of claim **1**, wherein the first value is generated to indicate a number of bits used to represent the second sub stream ID.

4. The method of claim **1**, wherein the second value is a predetermined value that represents a fixed number of the additional bit positions being included in the first network packet.

5. The method of claim **1**, wherein the control portion of the first network packet is generated to identify a purpose of the first network packet in relation to the stream.

6. The method of claim **1**, wherein the control portion of the first network packet further comprises a network communication type identifier.

7. The method of claim **1**, further comprising:

associating priority values with sub media streams of the stream;

generating a total priority value based on the priority values;

calculating desired transmission proportions corresponding to the sub media streams based on the priority values and the total priority value;

communicating a set of network communications comprising the first and second network packets over the sub media streams;

assessing sub media stream counts corresponding to the sub media streams based on the communicated set;

summing a total count based on the sub media stream counts;

determining actual transmission proportions corresponding to the sub media streams based on the sub media stream counts and the total count;

figuring ratios corresponding to the sub media streams based on the desired transmission proportions and the actual transmission proportions; and

storing the figured ratios.

8. The method of claim **7**, further comprising:

selecting a low-ratio sub media stream col responding to a lowest of the stored ratios;

queuing a set of real-time priority based network communications to be communicated over the sub media streams;

picking a low-ratio sub media stream network communication, associated with the low-ratio sub media stream, from the queued set; and

communicating the low-ratio sub media stream network communication over the low-ratio sub media stream.

9. One or more non-transitory computer storage media storing computer-useable instructions that, when used by one or more computing devices, cause the one or more imputing devices to perform operations comprising:

at a client system, establishing, based at least in part on cryptographic information in a predefined portion of a handshake network communication, a communication session with a server system for streaming one or more media streams from the server system to the client system;

receiving through the established communication session, as part of a stream of the one or more media streams, a plurality of network communications that includes a first network packet concatenated with a second network packet;

identifying a first sub stream identifier (ID) in a control portion of the first network packet;

identifying a second sub stream ID in a control portion of the second network packet of the plurality of network communications, wherein fewer bits are allocated for representing the second sub stream ID in the control portion of the second network packet than for representing the first sub stream ID in the control portion of the first network packet by a first value at bit positions of the control portion in the second network packet representing the second sub stream ID while a second value at the bit positions of the control portion in the first network packet to indicates that additional bit positions are included in the first network packet to represent the first sub stream ID;

processing the first network packet based on the first sub stream ID and the second network packet based on the second sub stream ID.

10. The one or more computer storage media of claim **9**, wherein the identifying the first sub stream ID comprises determining that the second value represents the first sub stream ID.

11. The one or more computer storage media of claim **9**, wherein the identifying the second sub stream ID comprises determining that the second value and a third value at the additional bits positions collectively represent the second sub stream ID.

12. The one or more computer storage media of claim **9**, further comprising identifying a purpose of the first network packet in relation to the stream from the control portion of the first network packet.

13. The one or more computer storage media of claim **9**, wherein the control portion of the first network packet further comprises a network communication type identifier.

14. The one or more computer storage media of claim **3**, further comprising:

receiving through the communication session, as part of the stream, values of parameters relating to a sub strewn, included in a first header portion of the first network packet;

storing the values of the parameters;

obtaining through the communication session, as part of the media stream, state information included in a control portion of the second network packet and a data payload included in the second network packet;

identifying, from the state information, a purpose of the second network packet in relation to the stream, and whether a second header portion of the second network packet includes one or more new values corresponding to one or more of the parameters;

updating, when the second header portion includes the one or more new values, one or more of the stored values based at least in part on the one or more new values; and

processing the data payload based at least in part on the identified purpose and the stored values of the parameters.

15. A media streaming system comprising:

a memory device;

a server system configured to execute instructions stored in the memory device that when executed perform operations comprising:

establishing, at the server system, based at least in part on cryptographic information placed in a predefined portion of a handshake network communication, a communication session with a client system for streaming one or more media streams from the server system to the client system;

processing a stream of the one of more streams, the processing comprising:

generating a first network packet of the stream, the first network packet having a first sub stream identifier (ID) in a control portion of the first network packet; and

generating a second network packet of the stream, the second network packet having a second sub stream ID in a control portion of the second network packet, wherein fewer bits are allocated for representing the second sub stream ID in the control portion of the second network packet than for representing the first sub stream ID in the control portion of the first network packet by storing a first value at bit positions of the control portion in the second network packet that represents the second sub stream ID while storing a second value at the bit positions of the control portion in the first network packet to indicate that additional bit positions are included in the first network packet to represent the first sub stream ID; and

streaming the processed stream to the client system via the established communication session by concatenating the first network packet and the second network packet.

16. The media streaming system of claim **15**, wherein the second value is generated to indicate a number of bits used to represent the first sub stream ID.

17. The media streaming system of claim **15**, wherein the first value is generated to indicate a number of bits used to represent the second sub stream ID.

18. The media streaming system of claim **15**, wherein the second value is a predetermined value that represents a fixed number of the additional bit positions being included in the first network packet.

19. The media steaming system of claim **15**, wherein the control portion of the first network packet is generated to identify a purpose of the first network packet in relation to the stream.

20. The media streaming system of claim **15**, wherein the control portion of the first network packet further comprises a network communication type identifier.

* * * * *